COMPARACIÓN DE MODELOS DE SINCRONIZACIÓN EN PROGRAMACIÓN PARALELA SOBRE CLUSTER DE MULTICORES

Autor: A.P.U. Enzo Rucci

Director: Ing. Armando E. De Giusti

Co-Director: Lic. Franco Chichizola

Tesina de grado - Licenciatura en Informática
Facultad de Informática - Universidad Nacional de La Plata

III-LIDI

Contexto

- Está enmarcada dentro de los proyectos acreditados por UNLP en el Programa de Incentivos
  - Arquitecturas multiprocesador distribuidas. Modelos, software de base y aplicaciones.
  - Procesamiento paralelo y distribuido. Fundamentos y aplicaciones en sistemas inteligentes y tratamiento de imágenes y video.
- Los resultados de esta investigación fueron publicados en el X Workshop de Procesamiento Paralelo y Distribuido realizado en el marco del XVI Congreso Argentino de Ciencias de la Computación (Octubre de 2010) bajo el título “Comparación de modelos de comunicación/sincronización en programación paralela sobre cluster de multicores”.

Objetivos

- Investigar modelos de sincronización para aplicaciones paralelas de alta complejidad computacional sobre cluster de multicores.
- Comparar diferentes técnicas de descomposición paralela y mapeo de tareas concurrentes a procesadores.
- Analizar alternativas de algoritmos paralelos para el alineamiento de secuencias considerando la arquitectura multiprocesador de soporte.
- Comparar las distintas soluciones implementadas, las cuales emplean sincronización por mensajes, por memoria compartida e híbrida, por medio de métricas de rendimiento tradicionales.

Agenda

- Motivación
  - Arquitecturas paralelas
  - Principios para el diseño de algoritmos paralelos
  - Evaluación de sistemas paralelos
  - Bioinformática
- Trabajo experimental / Resultados
  - Soluciones
  - Experimentos realizados y sus resultados
- Conclusiones
- Líneas de trabajo futuras

Agenda

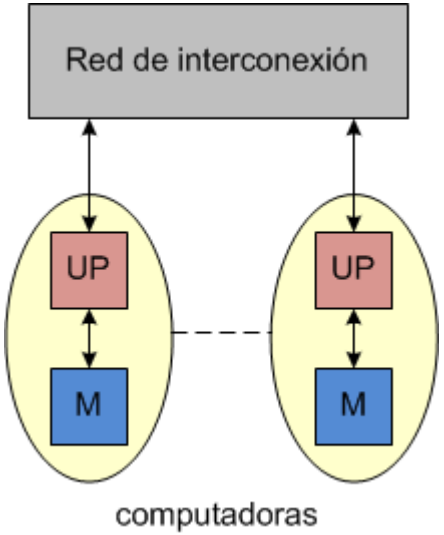
- Motivación
  - Arquitecturas paralelas
  - Principios para el diseño de algoritmos paralelos
  - Evaluación de sistemas paralelos
  - Bioinformática
- Trabajo experimental / Resultados
  - Soluciones
  - Experimentos realizados y sus resultados
- Conclusiones
- Líneas de trabajo futuras

Arquitecturas paralelas

- Plataforma paralela
  - Dos o más unidades de procesamiento vinculadas a partir de algún tipo de red de interconexión.
- Clasificación
  - Plataformas de memoria distribuida
  - Plataformas de memoria compartida
  - Plataformas de memoria combinada

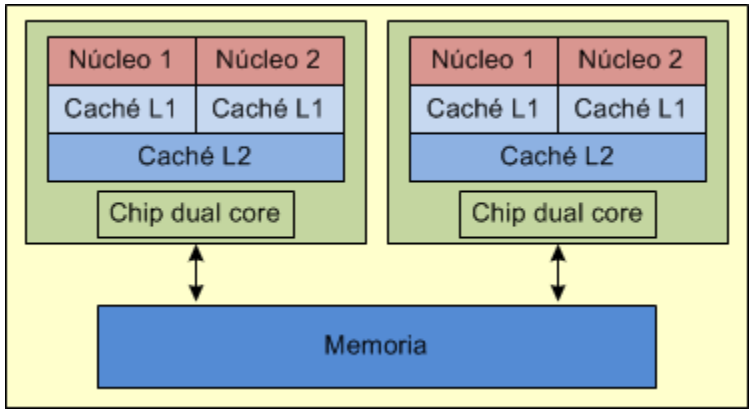
Plataformas de memoria distribuida

- Cluster
  - Colección de computadoras individuales interconectadas que trabajan en conjunto para resolver un problema determinado.



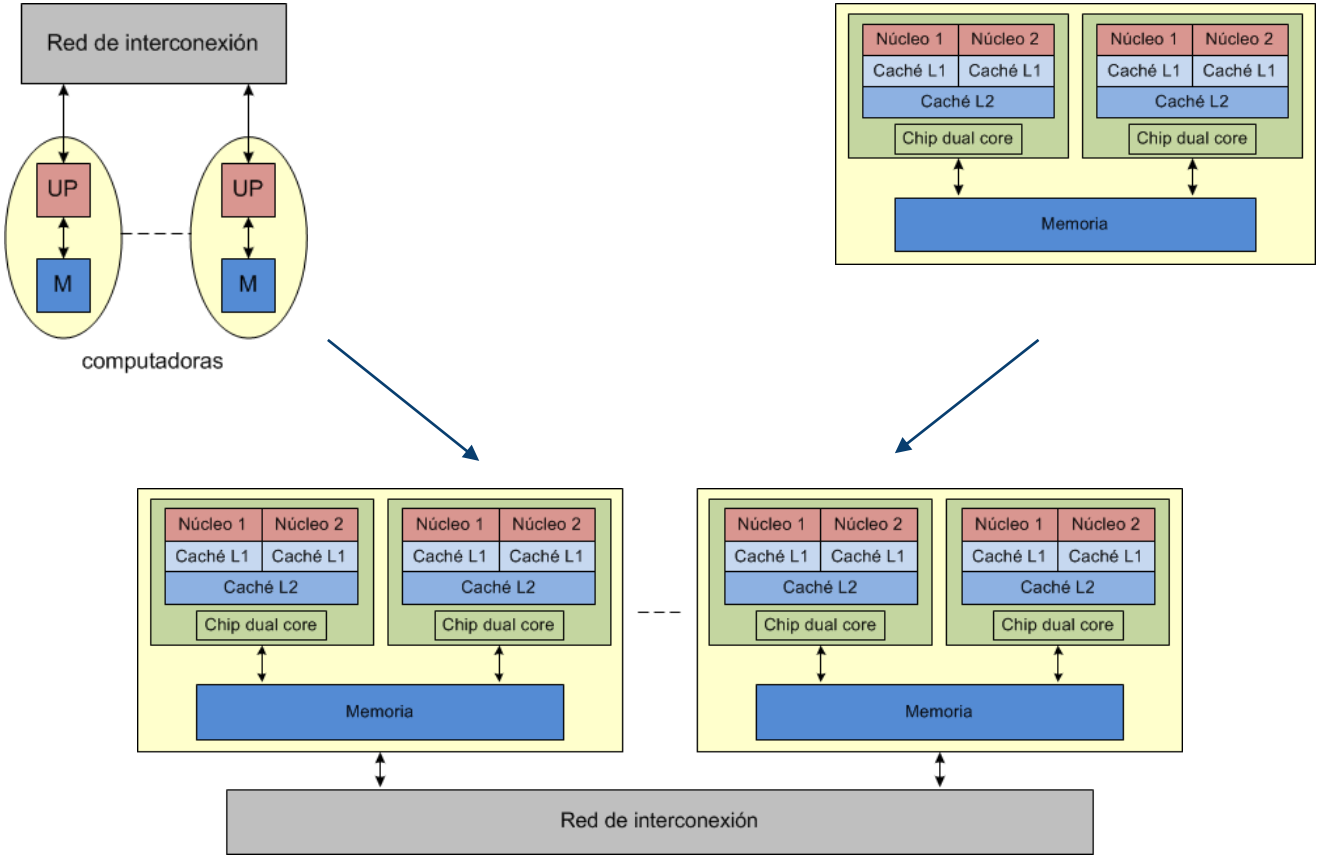
Plataformas de memoria compartida

- Multicore
  - Dos o más núcleos computacionales dentro de un mismo chip.



Plataformas de memoria combinada

- Cluster de multicores



Agenda

- Motivación
  - Arquitecturas paralelas
  - Principios para el diseño de algoritmos paralelos
  - Evaluación de sistemas paralelos
  - Bioinformática
- Trabajo experimental / Resultados
  - Soluciones
  - Experimentos realizados y sus resultados
- Conclusiones
- Líneas de trabajo futuras

Principios para el diseño de algoritmos paralelos

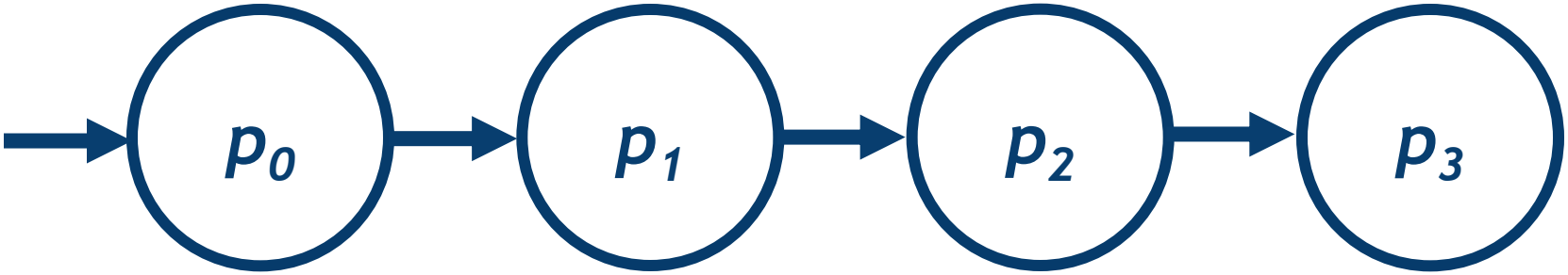
- Modelo de algoritmo paralelo
  - Permite estructurar la computación.
  - Técnica de descomposición + Técnica de mapeo + Estrategias para reducir overhead.
  - Diferentes clasificaciones
    - De acuerdo al patrón de interacción

Modelos de algoritmos paralelos de acuerdo al patrón de interacción

- Existen distintos modelos:
  - Modelo de datos paralelos
  - Modelo de grafo de tareas
  - Modelo *Bag of Tasks*
  - Modelo *Master-Slave*
  - Modelo productor-consumidor o pipeline
  - Modelo compuesto

Modelo pipeline

- Un flujo de datos es pasado a través de una sucesión de procesos, donde cada uno de ellos realiza alguna tarea sobre el mismo.
- La llegada de nuevos datos a un proceso dispara la ejecución de una nueva tarea.
- Diferentes configuraciones.



Modelos de programación según el espacio de direcciones

- Diversos lenguajes de programación y librerías han sido desarrollados para programación paralela explícita.
- Estos difieren principalmente en la manera en que el usuario ve el espacio de direcciones.
- Los modelos se dividen en:
  - Espacio de direcciones distribuido
  - Espacio de direcciones compartido
  - Espacio de direcciones híbrido

Espacio de direcciones distribuido

- Cada proceso cuenta con su memoria local.
- No existe una memoria compartida.
- Los procesos intercambian información enviando y recibiendo mensajes.
- Librería MPI
  - Define un estándar para el pasaje de mensajes.
  - Puede ser utilizado desde C y Fortran.
  - Define la sintaxis y la semántica de un conjunto de rutinas que resultan útiles para escribir programas con mensajes.

Espacio de direcciones compartido

- Todos los hilos comparten una única memoria.
- Los hilos intercambian información leyendo y escribiendo sobre variables que son compartidas.
- Librería Pthreads
  - Conjunto de funciones en el lenguaje C para la administración y sincronización de hilos.

Espacio de direcciones híbrido

- Combina características de los dos anteriores.
- Suele emplearse en arquitectura híbridas.
- Las tareas se comunican:
  - leyendo y escribiendo variables que son compartidas.
  - enviando y recibiendo mensajes.

Agenda

- Motivación
  - Arquitecturas paralelas
  - Principios para el diseño de algoritmos paralelos
  - Evaluación de sistemas paralelos
  - Bioinformática
- Trabajo experimental / Resultados
  - Soluciones
  - Experimentos realizados y sus resultados
- Conclusiones
- Líneas de trabajo futuras

Evaluación de sistemas paralelos

- Métricas de rendimiento
  - Tiempo de ejecución
    - Secuencial (T_s) y paralelo (T_p)
  - Speedup

$$S = \frac{T_s}{T_p}$$

  - Eficiencia

$$E = \frac{S}{p}$$

Fuentes de overhead

- Interacción entre procesos
- Ocio
- Exceso de cómputo

Agenda

- Motivación
  - Arquitecturas paralelas
  - Evaluación de sistemas paralelos
  - Principios para el diseño de algoritmos paralelos
  - Bioinformática
- Trabajo experimental / Resultados
  - Soluciones
  - Experimentos realizados y sus resultados
- Conclusiones
- Líneas de trabajo futuras

Bioinformática

- Surge a partir del desarrollo de técnicas que permiten descifrar la información que contiene el ADN.
- Busca no sólo adquirir, almacenar y organizar la información biológica que contiene la molécula de ADN, sino también analizar e interpretar esos datos.
- ¿Por qué es importante?

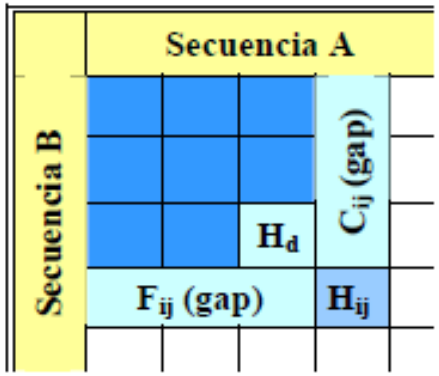
Alineamiento de secuencias

- Es el centro de todas las operaciones y análisis en el mundo bioinformático, tanto para la búsqueda de patrones entre secuencias de aminoácidos y nucleótidos, como para la búsqueda de relaciones filogenéticas entre organismos.
- Permite responder una serie de preguntas para la biología:
  - ¿Qué tan parecidas son dos secuencias ya conocidas?
  - ¿A qué puede llegar a parecerse una secuencia desconocida?
- El alineamiento puede realizarse de dos formas:
  - Global
  - Local

Algoritmo Smith-Waterman

- En 1981, Smith y Waterman desarrollaron un método para encontrar regiones comunes de similitud.
- El método Smith-Waterman ha servido como base para el desarrollo de otros algoritmos posteriores e incluso se lo utiliza como medida de referencia para diferentes técnicas de alineamiento.

Algoritmo Smith-Waterman



$$H_{ij} = \max \begin{cases} 0 \\ H_{i-1,j-1} + V(a_i, b_j) \\ C_{ij} \\ F_{ij} \end{cases}$$

- $A = a_1 a_2 a_3 \ldots a_M$ y $B = b_1 b_2 b_3 \ldots b_N$
- Matriz H de $(N+1)$x$(M+1)$
- Inicializar con 0's la fila 0 y la columna 0.

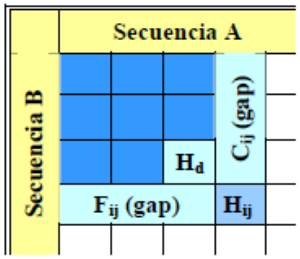
- $V(a_i, b_j)$ es la función de coincidencias
- C_{ij} es el puntaje considerando un *gap* en la columna j
- F_{ij} es el puntaje considerando un *gap* en la fila i

- Obtener puntaje de similitud G
- (Opcional) Retroceso en diagonal desde G

Agenda

- Motivación
  - Arquitecturas paralelas
  - Evaluación de sistemas paralelos
  - Principios para el diseño de algoritmos paralelos
  - Bioinformática
- Trabajo experimental / Resultados
  - Soluciones
  - Experimentos realizados y sus resultados
- Conclusiones
- Líneas de trabajo futuras

Alineamiento de secuencias de ADN



$+$ $O(N \text{x} M)$ $+$ $AAGC\ldots A$

10^9

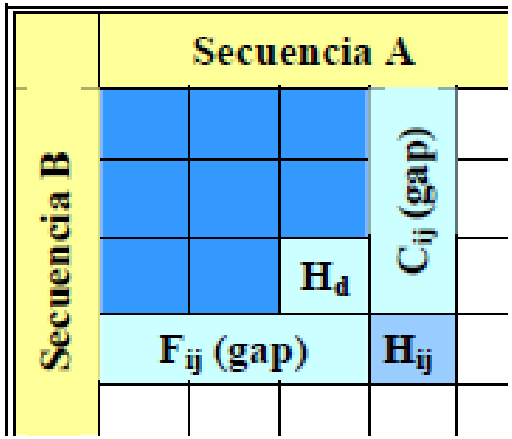
Estudiar paralelización del problema

Agenda

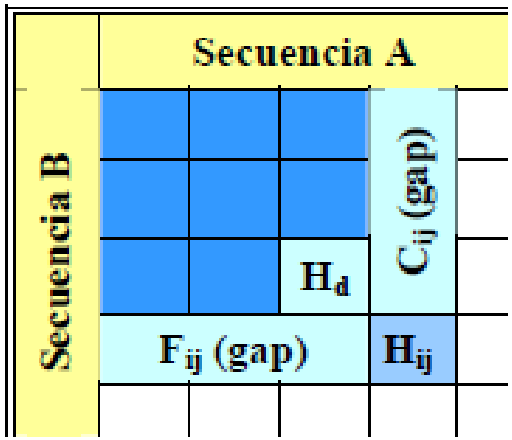
- Motivación
  - Arquitecturas paralelas
  - Evaluación de sistemas paralelos
  - Principios para el diseño de algoritmos paralelos
  - Bioinformática
- Trabajo experimental / Resultados
  - Soluciones
  - Experimentos realizados y sus resultados
- Conclusiones
- Líneas de trabajo futuras

Soluciones

- Sólo se calcula el puntaje de similitud entre dos secuencias y no se realiza el retroceso.
  - No es necesario almacenar la matriz *H*.



Solución secuencial

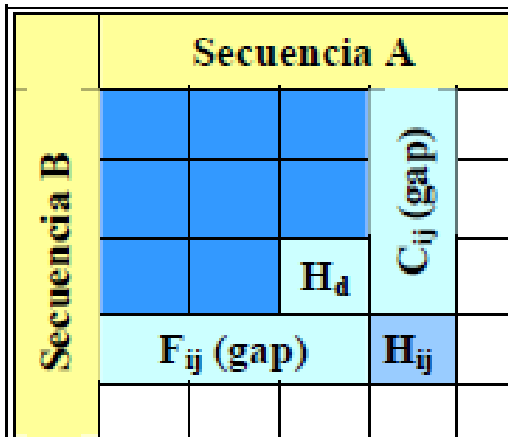


- Un vector *h* de longitud *M*+1 que mantiene en cada posición el valor obtenido en la última fila procesada sobre esa columna.

$$h_k = \begin{cases} H_{i,k} & k < j-1 \\ H_{i-1,k} & k \geq j-1 \end{cases}$$

- Un elemento *e* para guardar en forma temporal el último valor calculado en la fila que se está procesando.

Solución secuencial

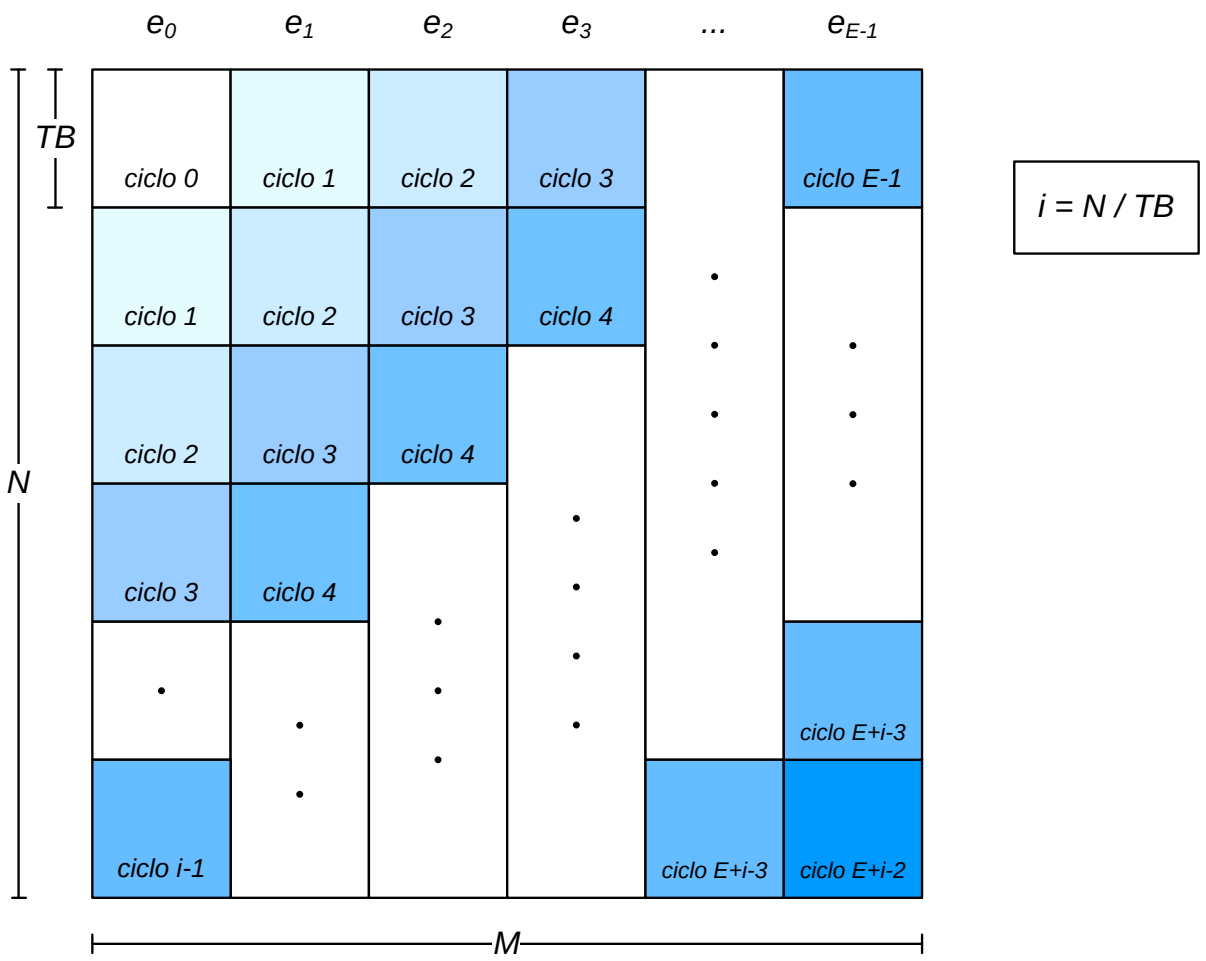


- Un vector *c* de longitud *M*+1 que mantiene en cada posición el máximo puntaje considerando un *gap* en esa columna.

$$c_k = \begin{cases} C_{ik} & k < j \\ C_{i-1,k} & k \geq j \end{cases}$$

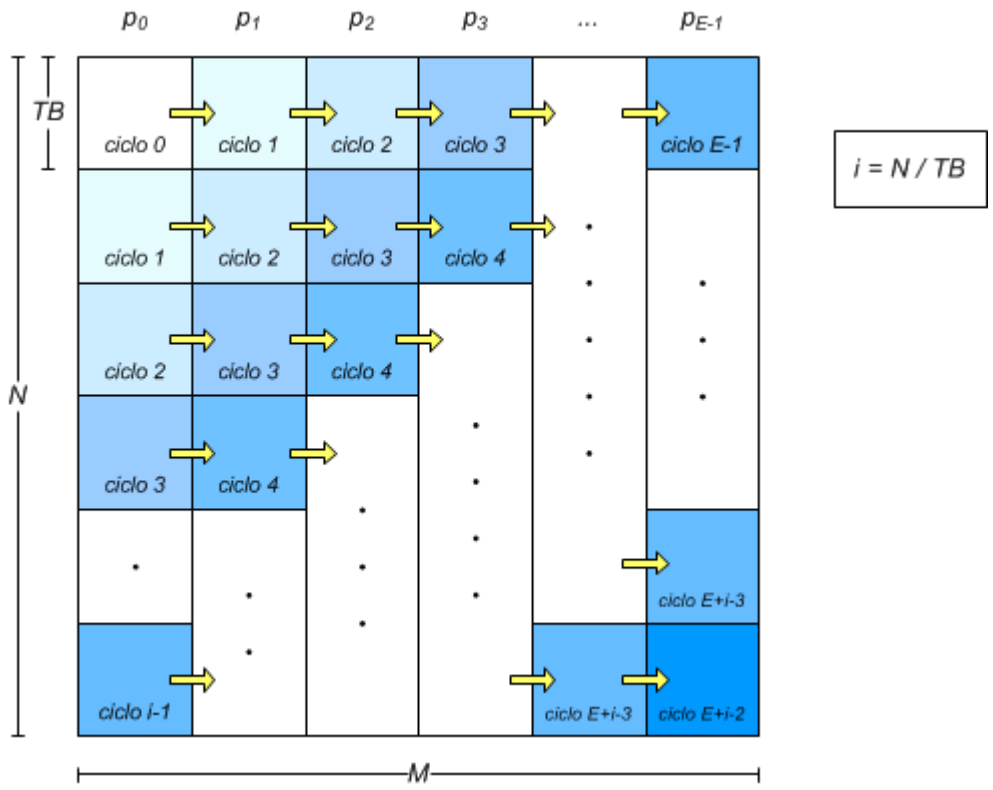
- Un elemento *f* que mantiene el máximo puntaje considerando un *gap* en la fila que se está procesando.

Solución paralela general



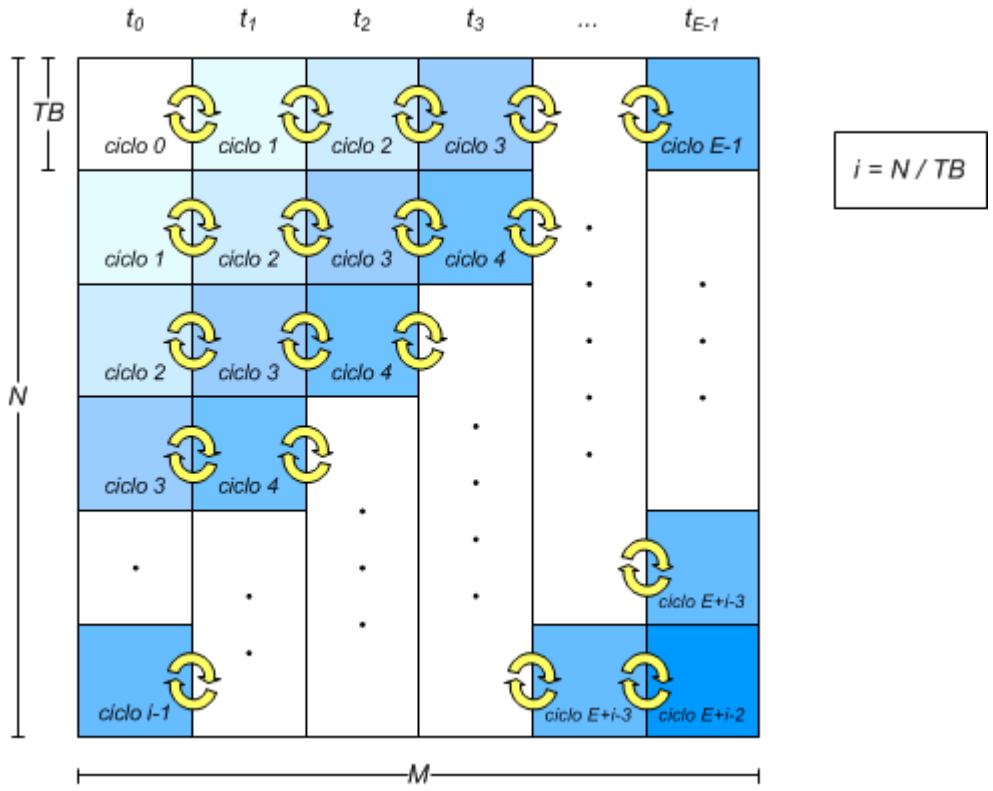
Solución paralela general

- Modelos de comunicación/sincronización
  - Pasaje de mensajes

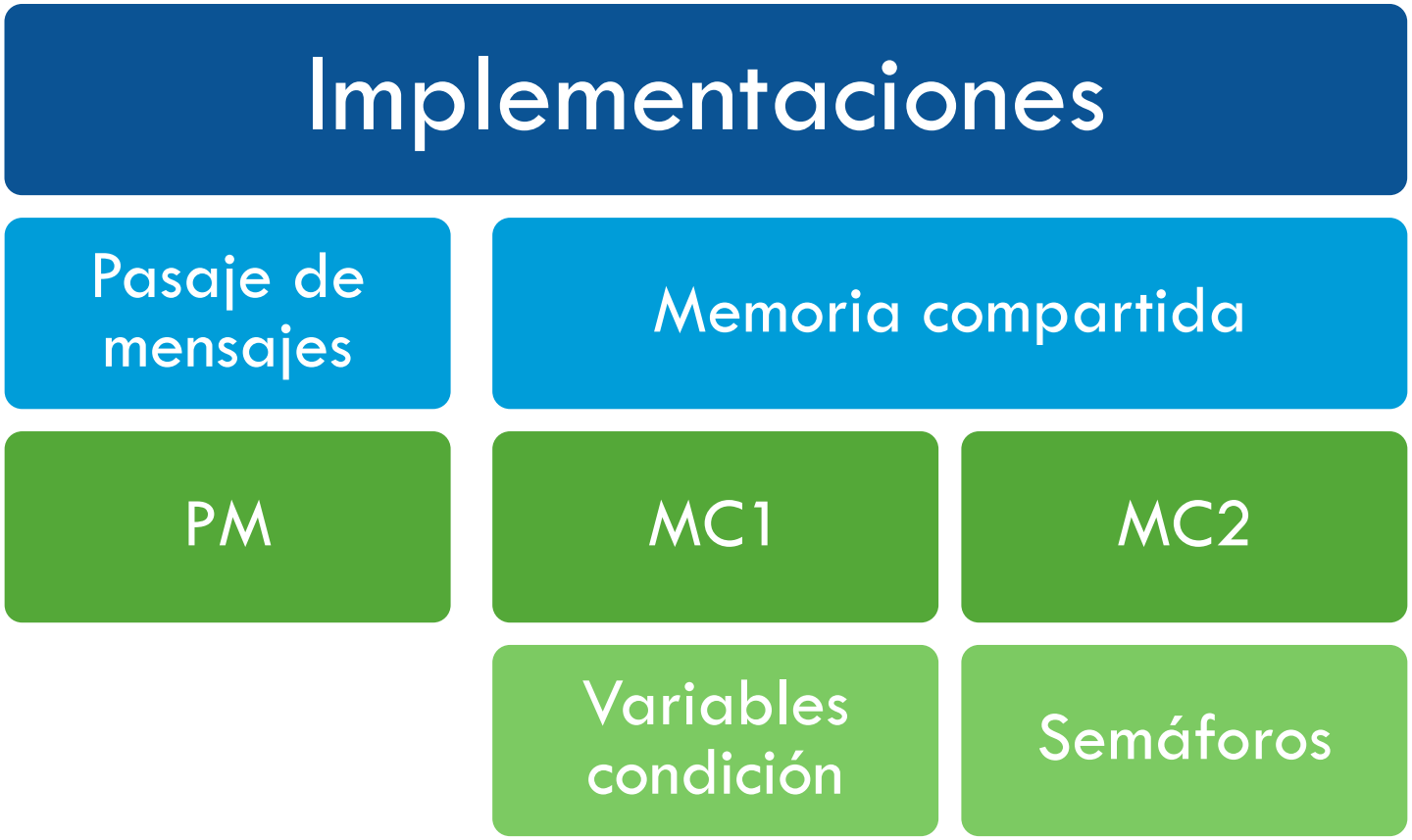


Solución paralela general

- Modelos de comunicación/sincronización
  - Memoria compartida



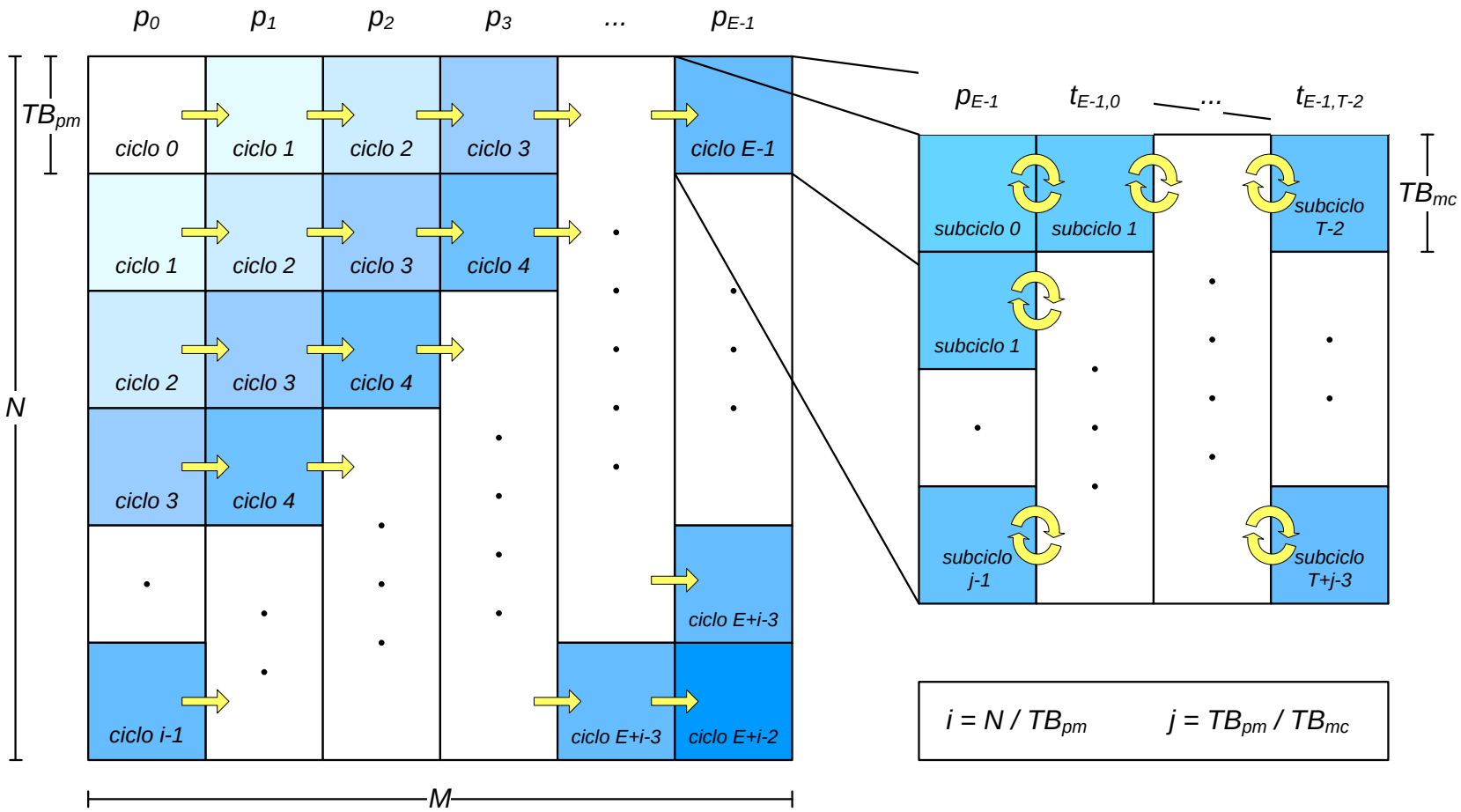
Solución paralela general



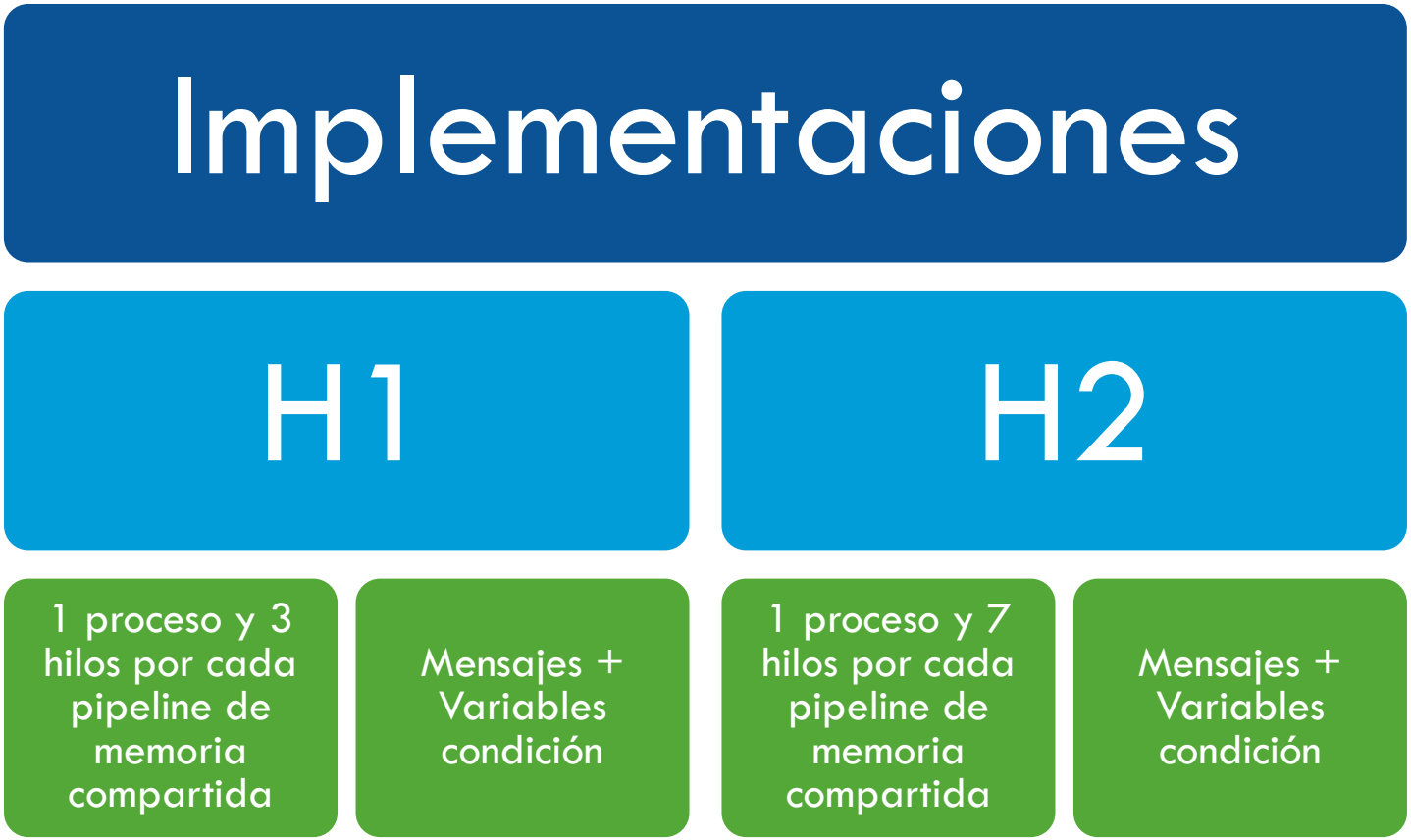
Soluciones paralelas híbridas

- Al utilizar como soporte una arquitectura híbrida se debe tener en cuenta al momento de desarrollar una solución paralela los diferentes niveles de memoria (entre núcleos de un mismo nodo) y la red de interconexión (entre núcleos de diferentes nodos). Esto lleva a plantear soluciones que combinan el uso de pasaje de mensajes con memoria compartida.

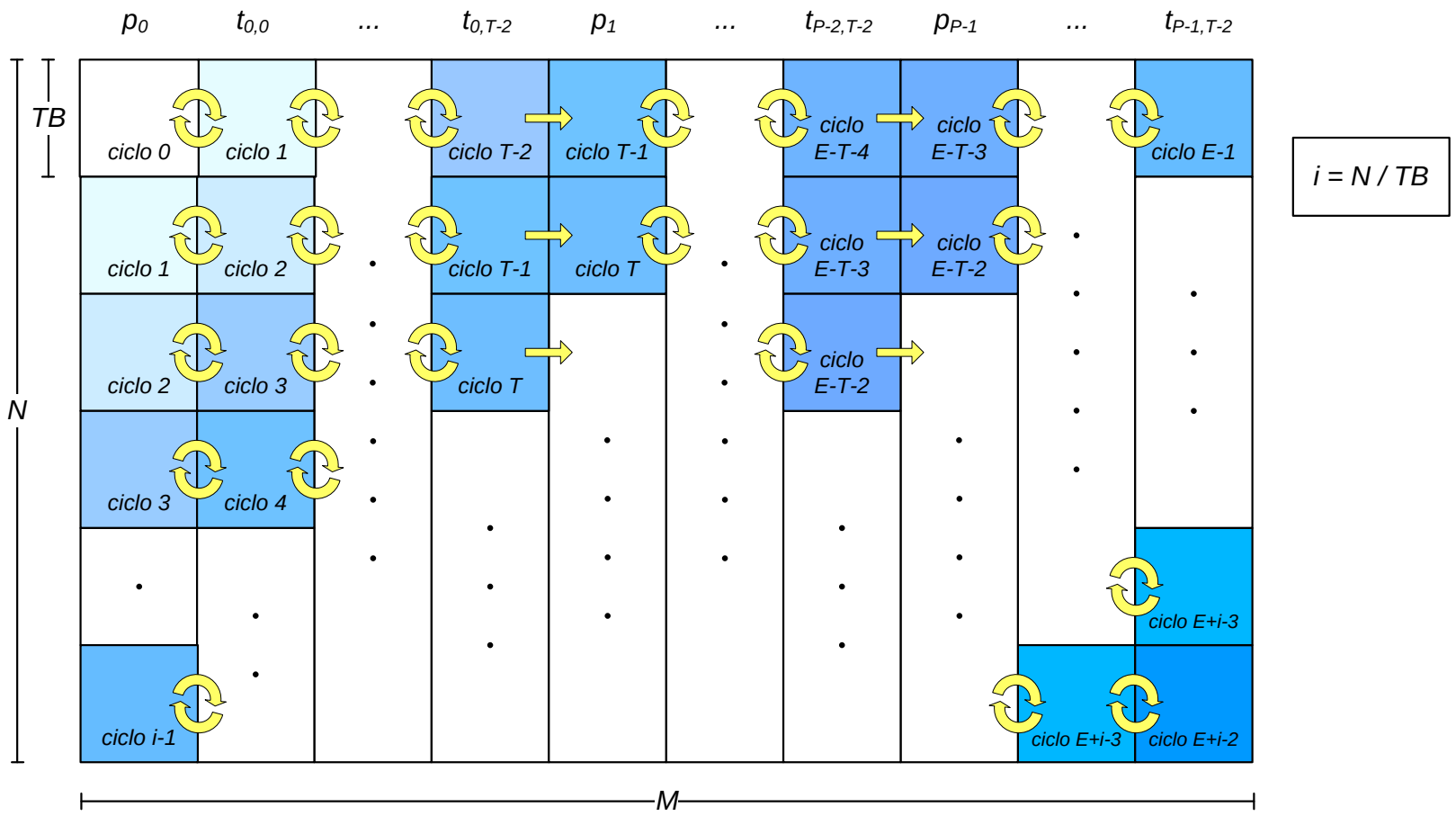
Solución paralela híbrida 1



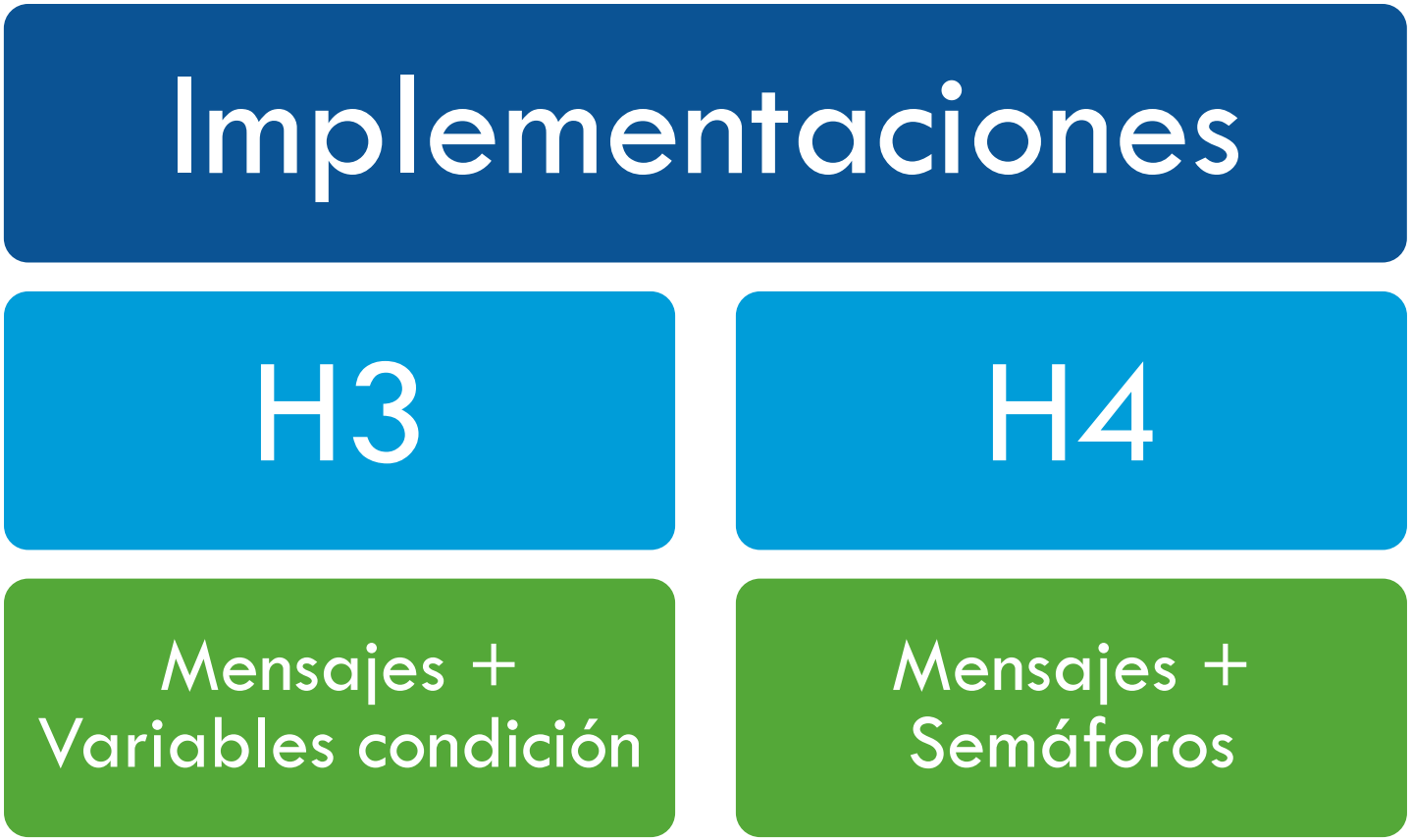
Solución paralela híbrida 1



Solución paralela híbrida 2



Solución paralela híbrida 2



Agenda

- Motivación
  - Arquitecturas paralelas
  - Evaluación de sistemas paralelos
  - Principios para el diseño de algoritmos paralelos
  - Bioinformática
- Trabajo experimental / Resultados
  - Soluciones
  - Experimentos realizados y sus resultados
- Conclusiones
- Líneas de trabajo futuras

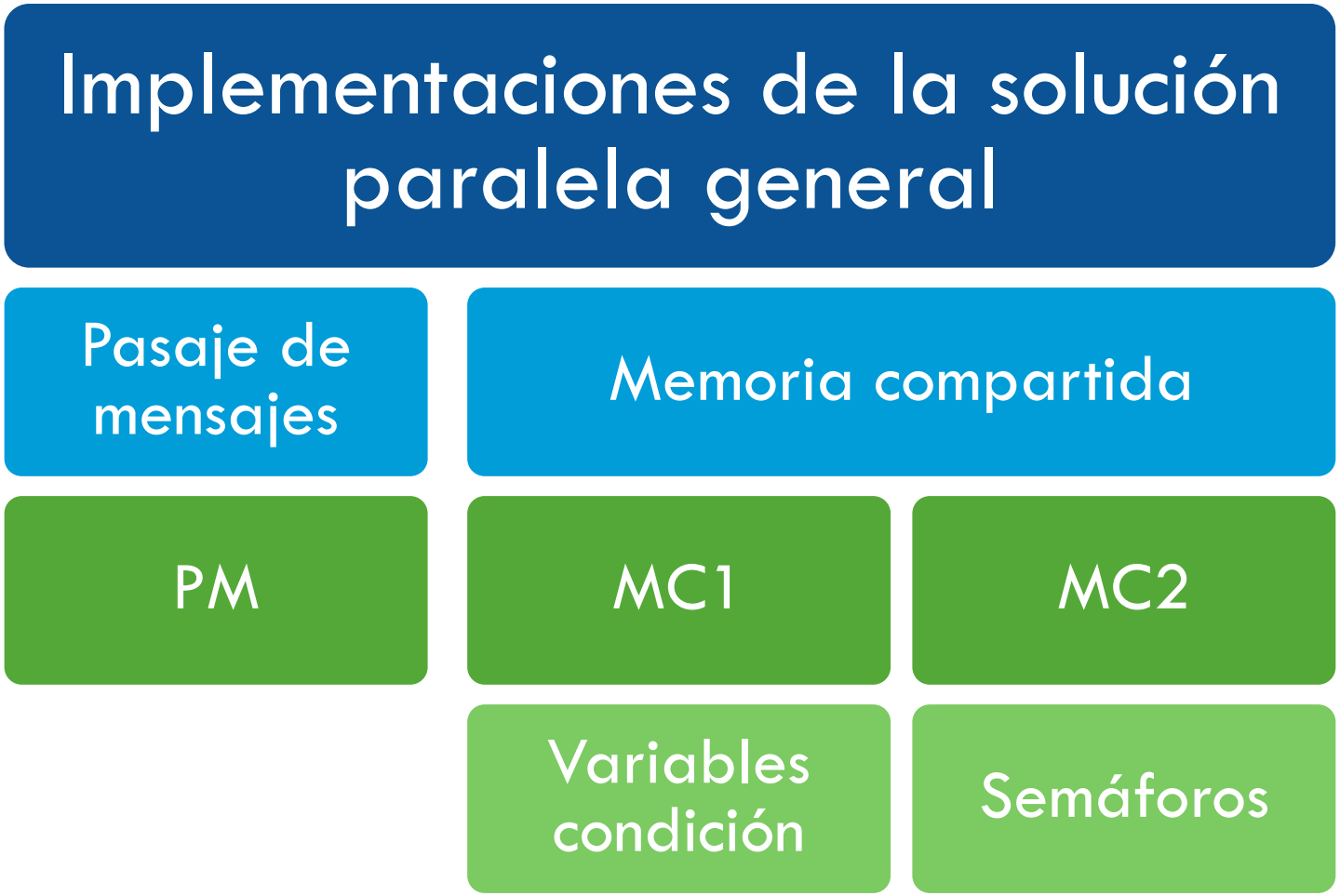
Experimentos realizados y sus resultados

- Lenguaje C + OpenMPI + Pthreads
- Blade de 8 hojas. Cada hoja:
  - 2 procesadores quad core Intel Xeon e5405 de 2.0Ghz.
  - 2 Gb de RAM (compartida entre procesadores) y caché L2 de 2x6Mb entre par de cores.
- 2 fases de pruebas

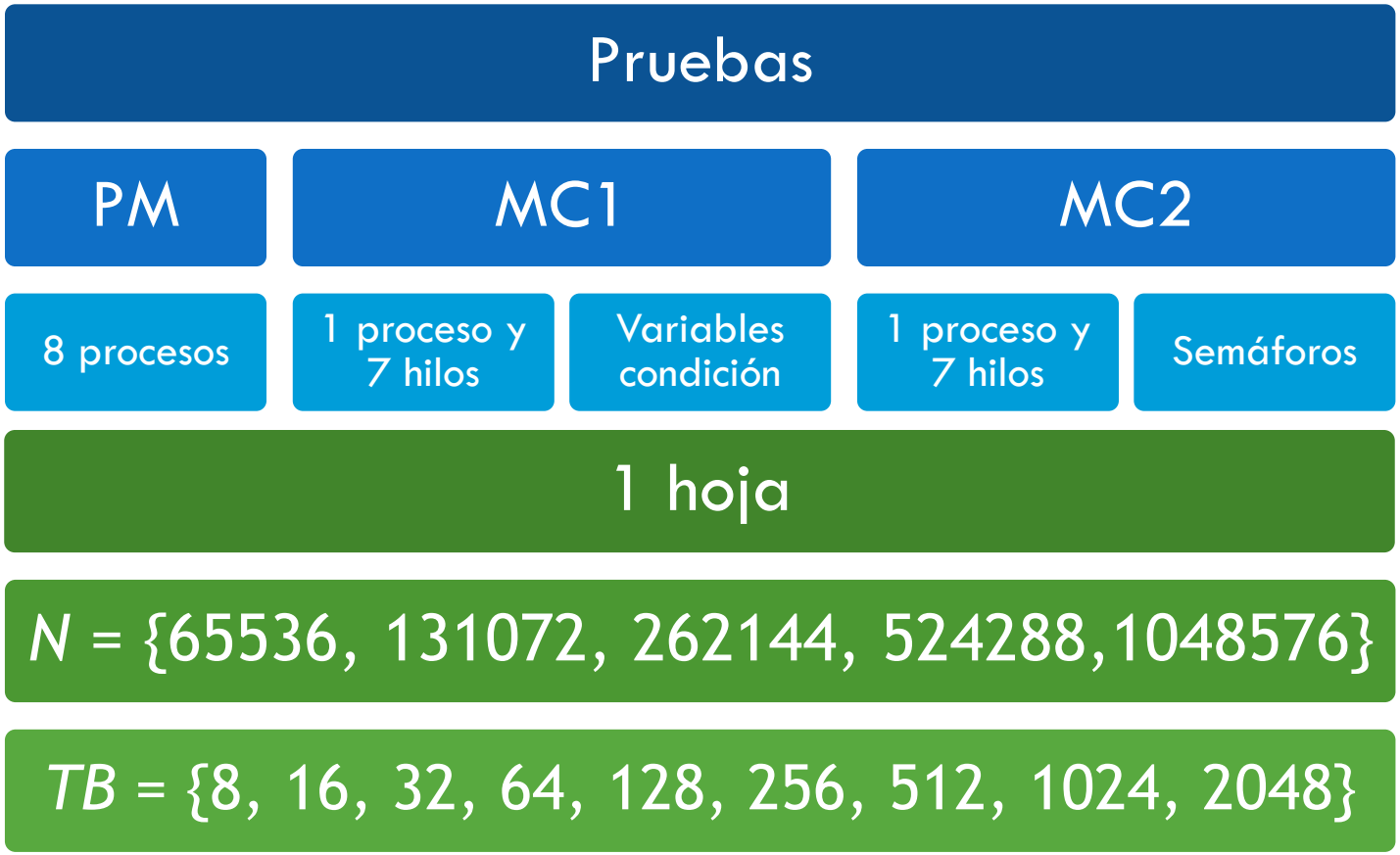
Primera fase de prueba

- El objetivo fue determinar de forma empírica los tamaños de bloques adecuados para los algoritmos paralelos puros y en base a eso determinar los correspondientes para los algoritmos híbridos.

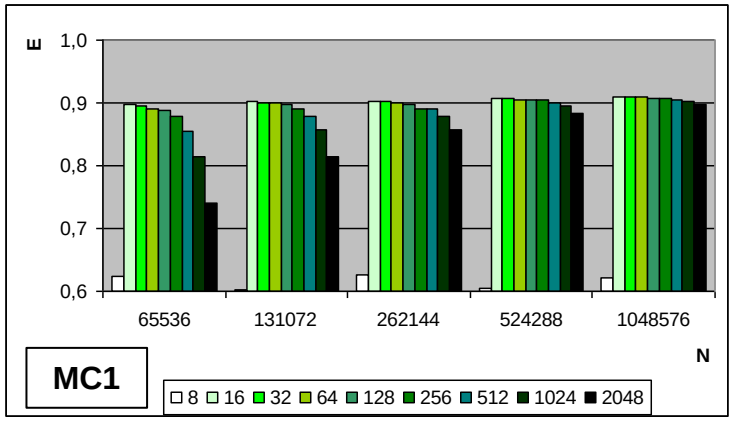
Primera fase de prueba

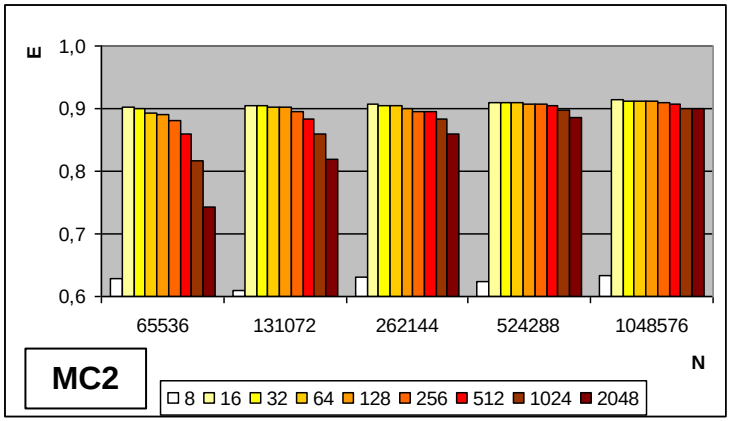


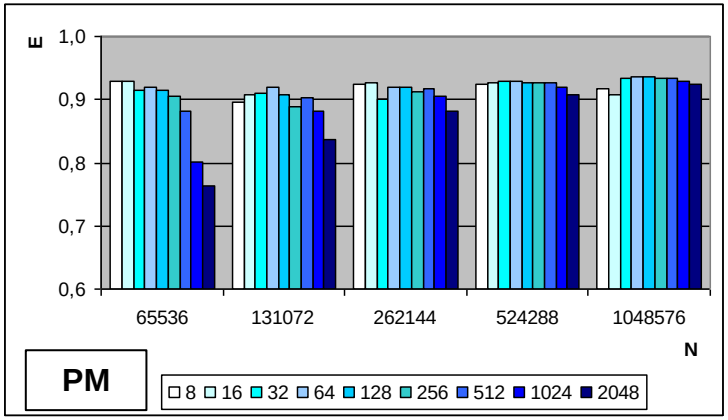
Primera fase de prueba



Resultados de la primera fase de prueba

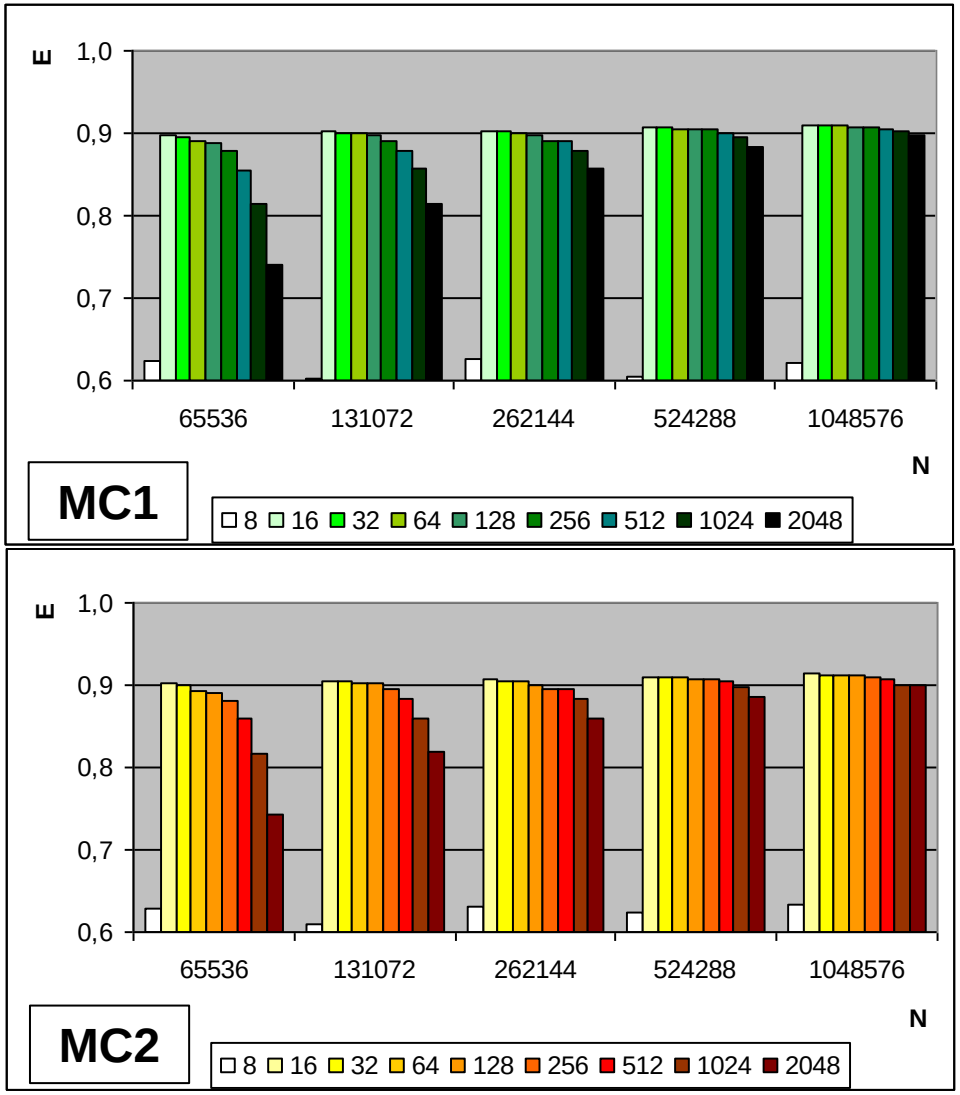






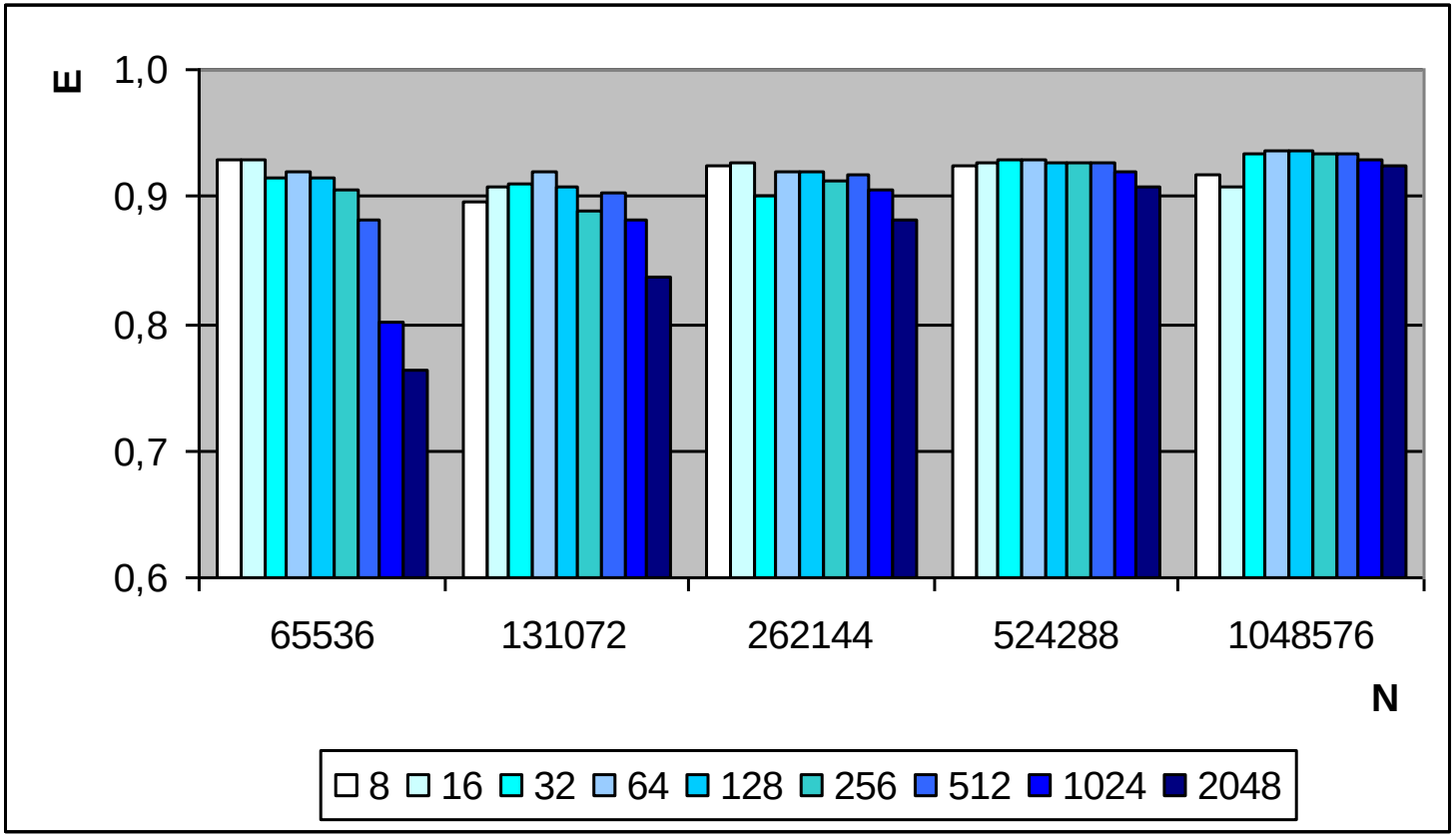
Eficiencia lograda por los algoritmos *MC1*, *MC2* y *PM* para diferentes tamaños de secuencia (N) y de bloques de datos utilizando una hoja (ocho núcleos) de la arquitectura.

Resultados de la primera fase de prueba



Eficiencia lograda por los algoritmos *MC1* y *MC2* para diferentes tamaños de secuencia (N) y de bloques de datos utilizando una hoja (ocho núcleos) de la arquitectura.

Resultados de la primera fase de prueba



Eficiencia lograda por el algoritmo *PM* para diferentes tamaños de secuencia (N) y de bloques de datos utilizando una hoja (ocho núcleos) de la arquitectura.

Primera fase de prueba

Pruebas

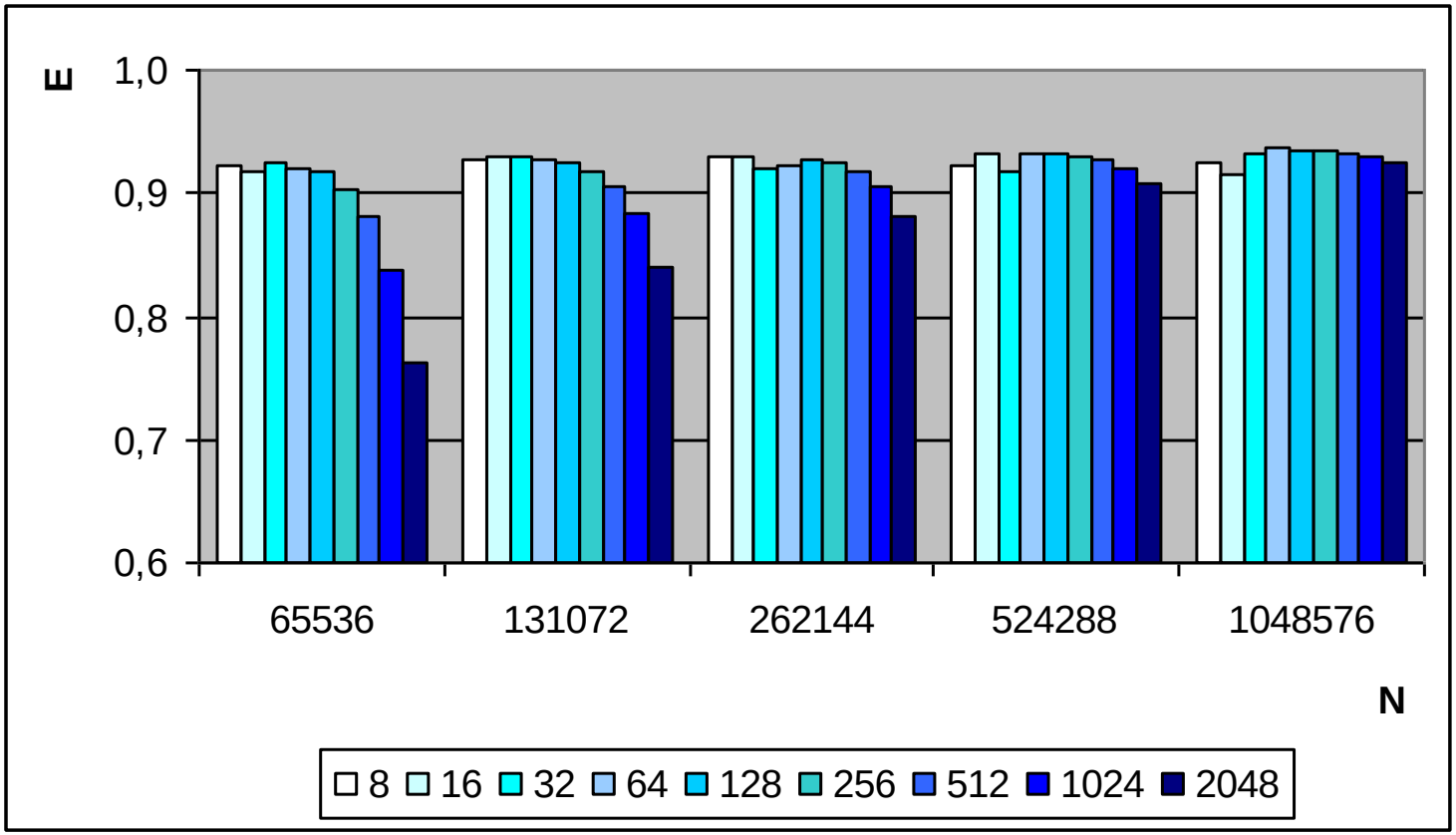
PM

1 núcleo de cada una de las 8 hojas

$N = \{65536, 131072, 262144, 524288, 1048576\}$

$TB = \{8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$

Resultados de la primera fase de prueba



Eficiencia lograda por el algoritmo *PM* para diferentes tamaños de secuencia (N) y de bloques de datos utilizando un núcleo de cada una de las ocho hojas de la arquitectura.

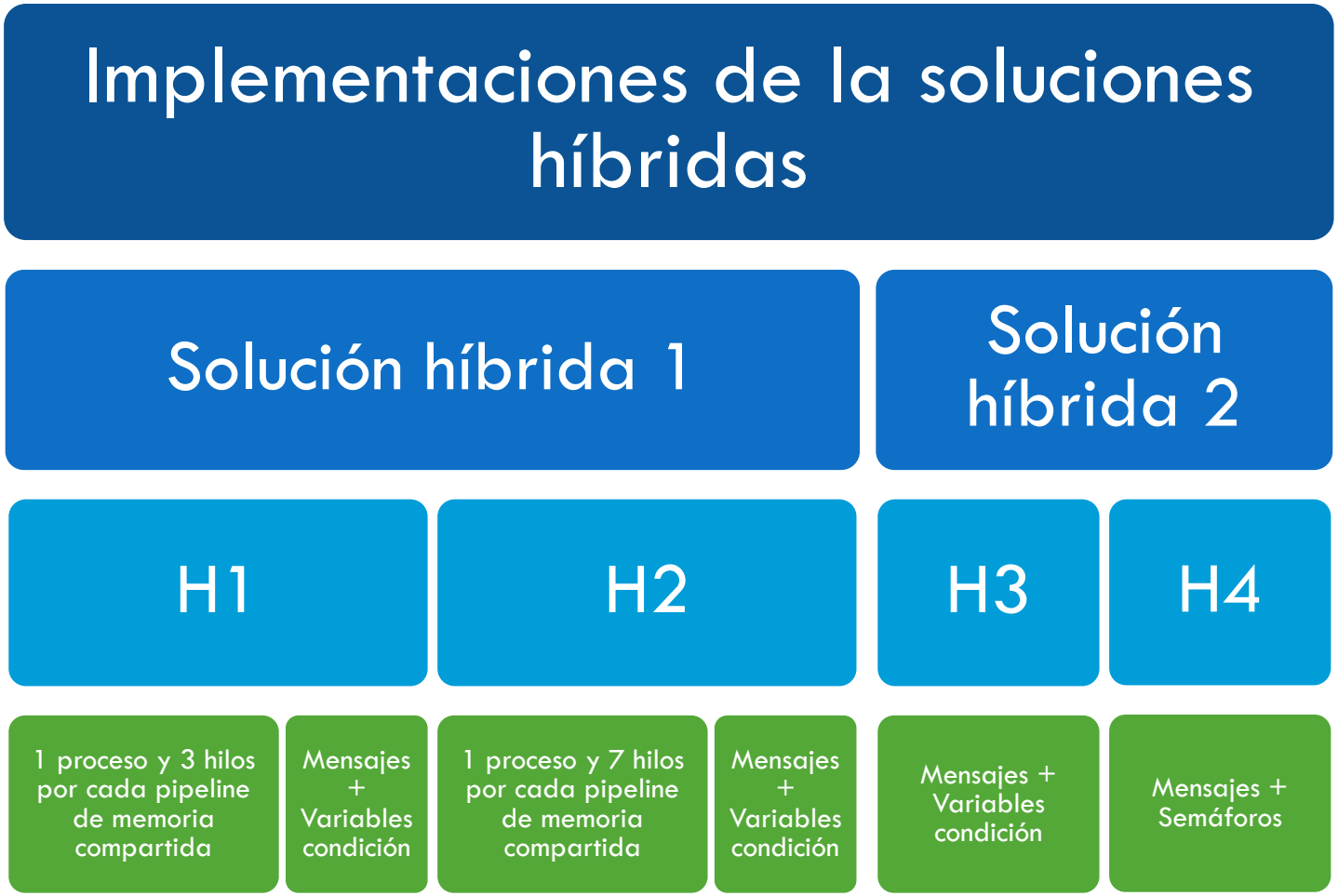
Resultados de la primera fase de prueba

Tamaño de bloque óptimo		
MC1	MC2	PM
16	16	TB < 512

Segunda fase de prueba

- El objetivo fue analizar y comparar el comportamiento de los algoritmos híbridos con el del algoritmo que solo emplea pasaje de mensajes.

Segunda fase de prueba



Segunda fase de prueba

Pruebas

PM | H3 | H4 | H1 | H2

TB = {8, 16, 32, 64, 128, 256, 512, 1024, 2048}

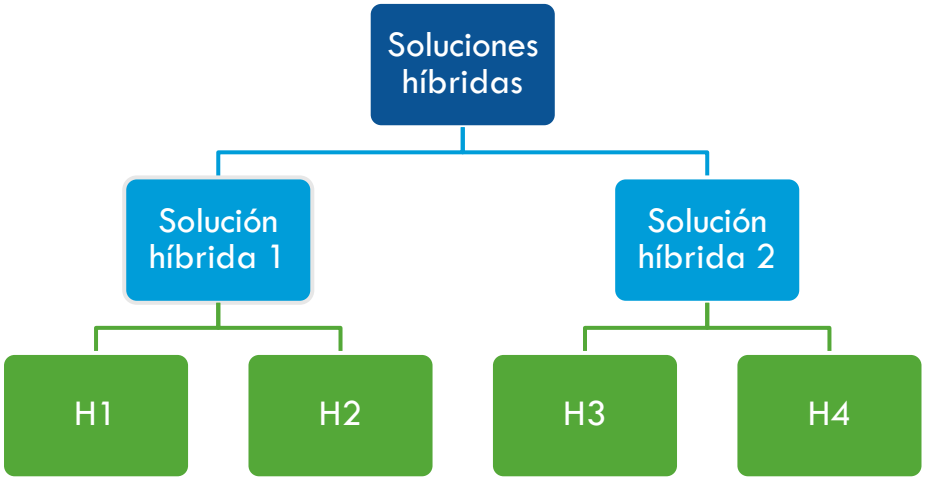
TB_{mc} = 16 y TB_{pm} = {128, 256, 512, 1024, 2048}

N = {65536, 131072, 262144, 524288,1048576}

4 y 8 hojas

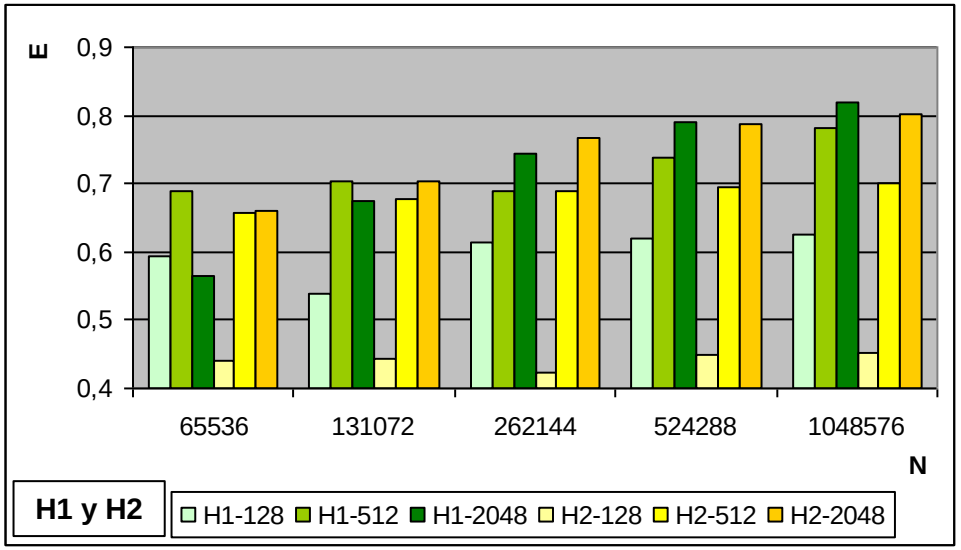
Resultados de la segunda fase de prueba

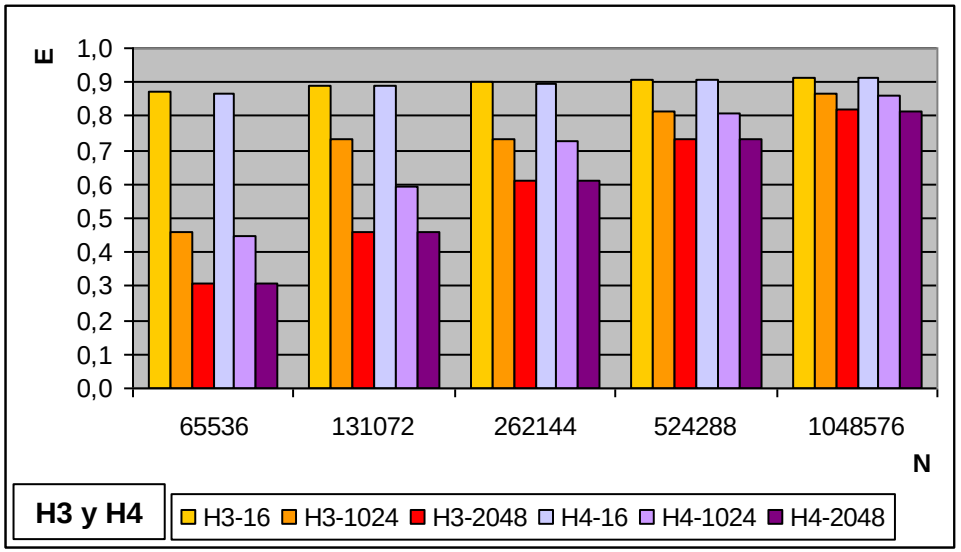
- Los algoritmos híbridos pueden ser aparejados de acuerdo a su esquema de resolución.



- Se procede a comparar los algoritmos híbridos según las agrupaciones realizadas, para luego analizar el mejor de cada una de ellas junto al algoritmo que sólo emplea pasaje de mensajes.

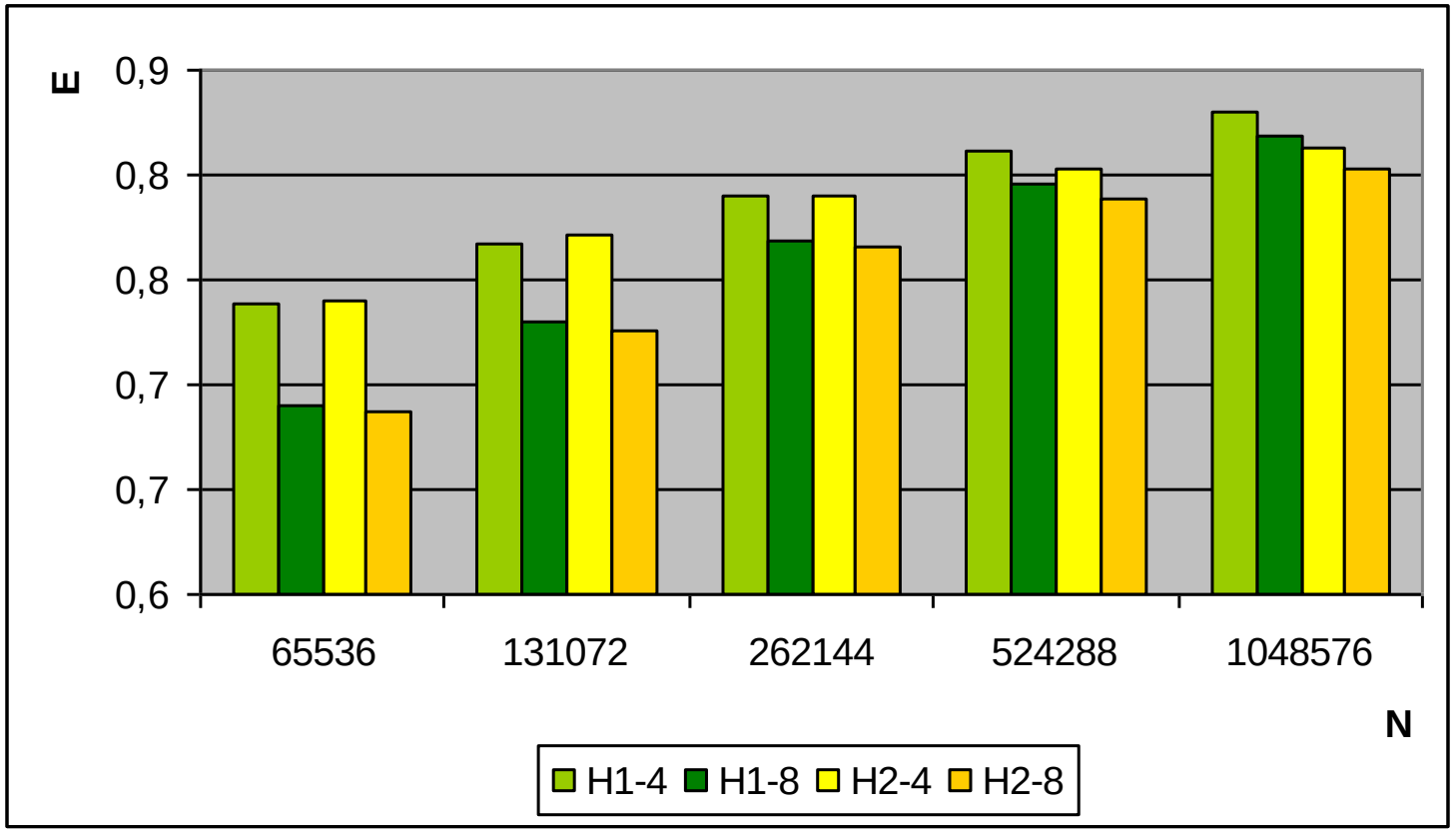
Resultados de la segunda fase de prueba





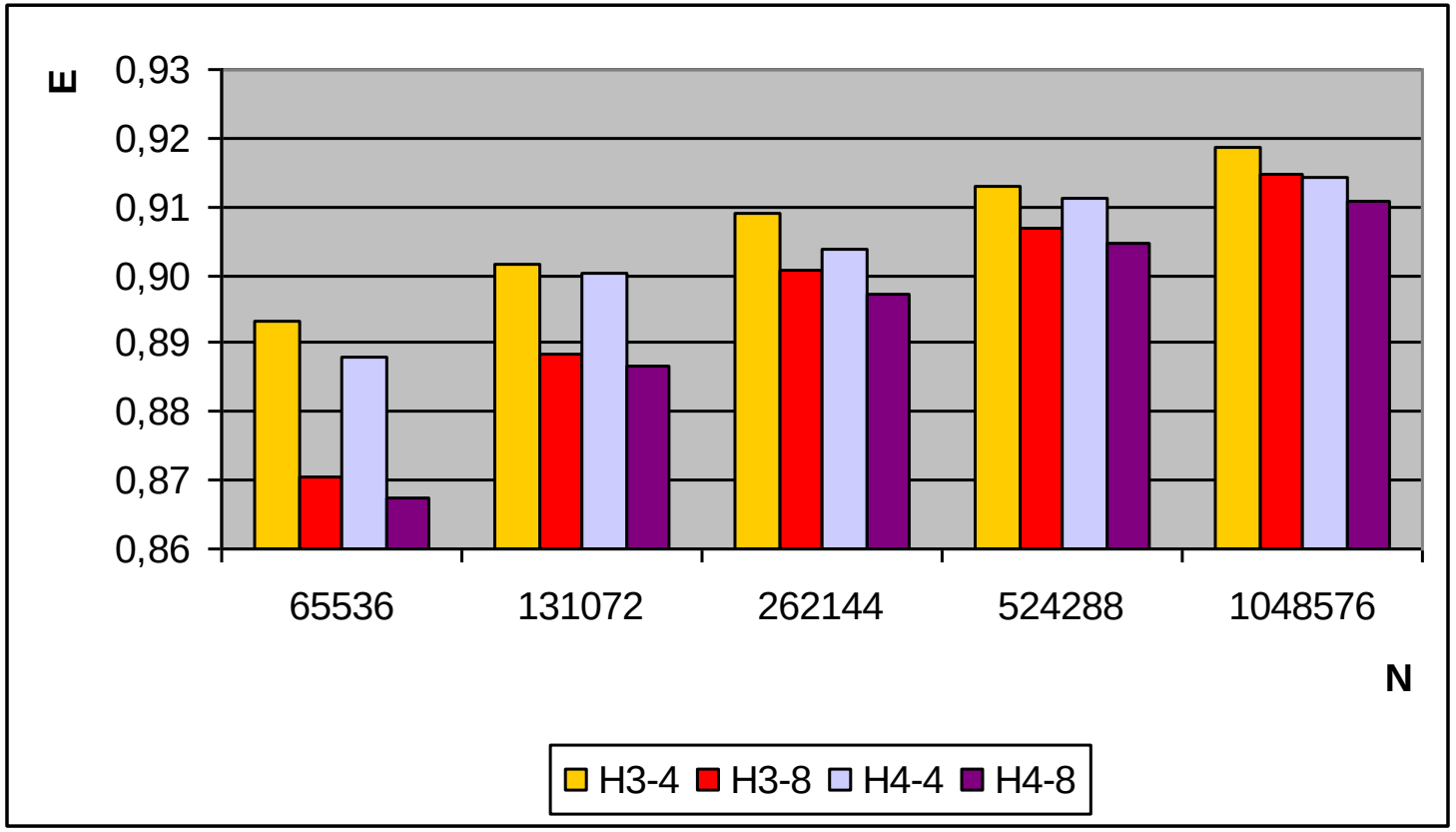
Eficiencia lograda por los algoritmos *H1*, *H2*, *H3* y *H4* para diferentes tamaños de secuencia (N) y de bloques de datos usando 64 núcleos.

Resultados de la segunda fase de prueba



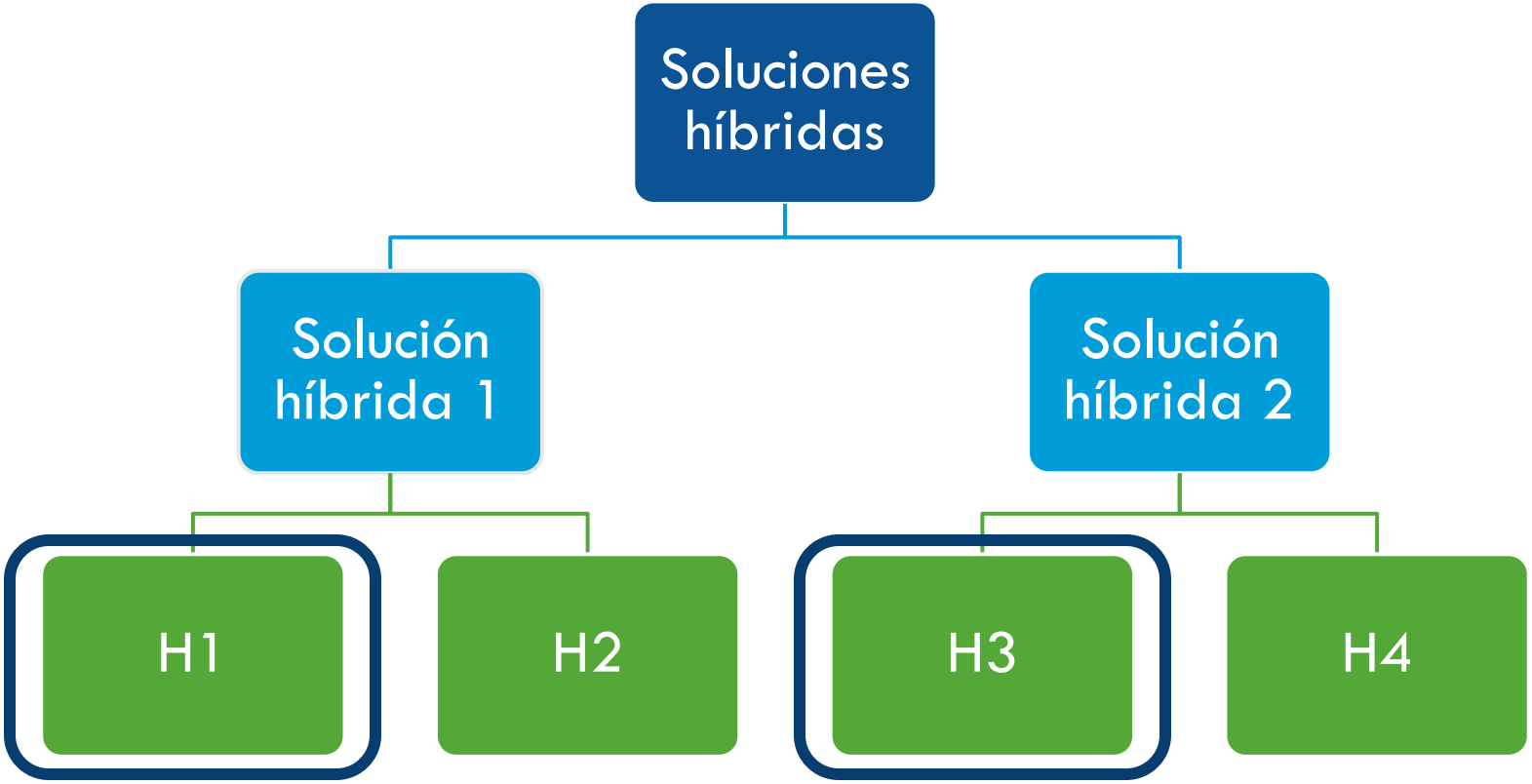
Resumen de la mejor eficiencia lograda por los algoritmos *H1* y *H2* para diferentes tamaños de secuencia (N) con cuatro y ocho hojas de la arquitectura.

Resultados de la segunda fase de prueba

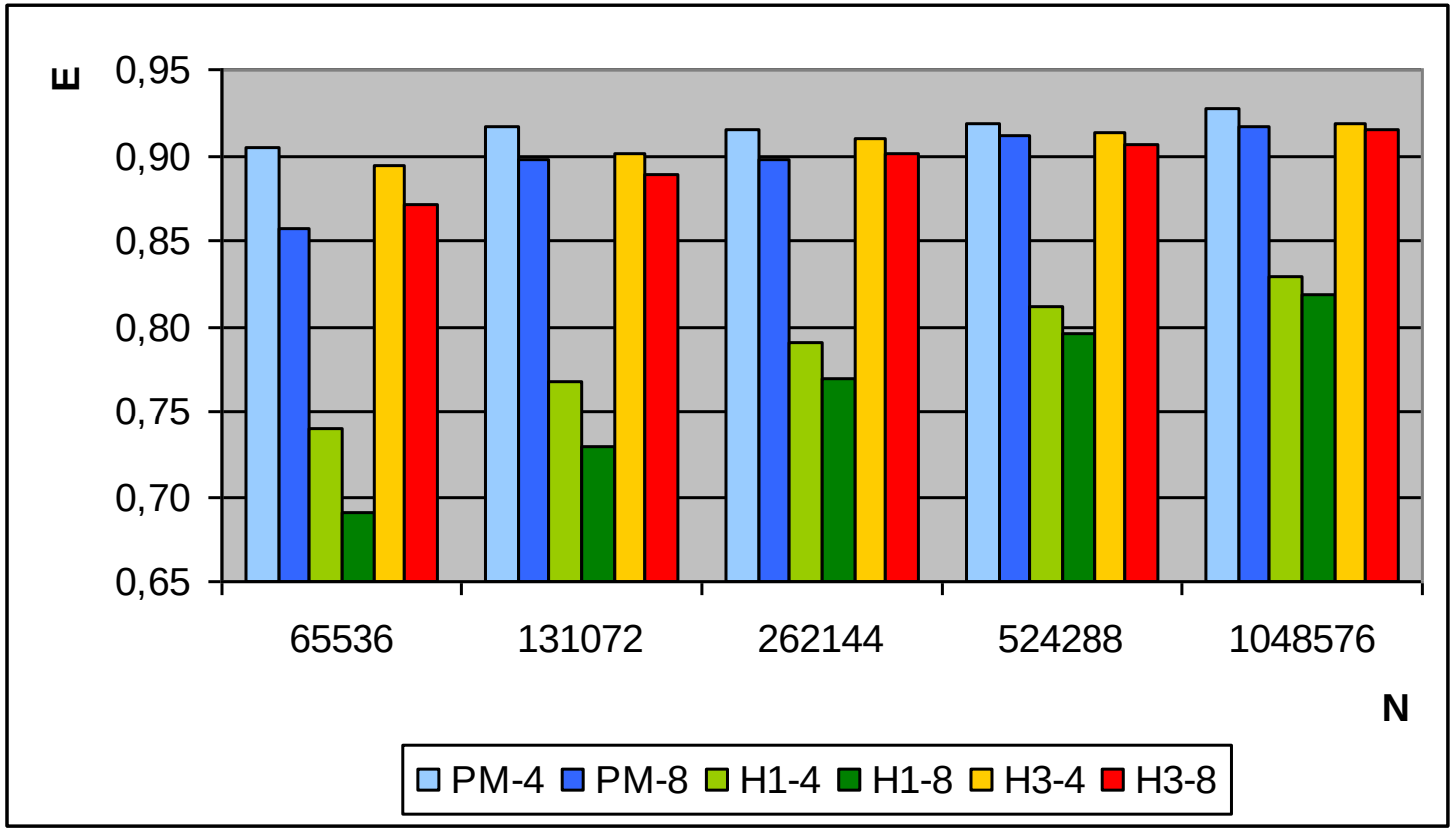


Resumen de la mejor eficiencia lograda por los algoritmos *H3* y *H4* para diferentes tamaños de secuencia (N) con cuatro y ocho hojas de la arquitectura.

Resultados de la segunda fase de prueba



Resultados de la segunda fase de prueba



Resumen de la mejor eficiencia lograda por los algoritmos *PM*, *H1* y *H3* para diferentes tamaños de secuencia (N) con cuatro y ocho hojas de la arquitectura.

Agenda

- Motivación
  - Arquitecturas paralelas
  - Evaluación de sistemas paralelos
  - Principios para el diseño de algoritmos paralelos
  - Bioinformática
- Trabajo experimental / Resultados
  - Soluciones
  - Experimentos realizados y sus resultados
- Conclusiones
- Líneas de trabajo futuras

Conclusiones

- En este trabajo se paraleliza el algoritmo Smith-Waterman mediante un esquema de pipeline debido a las dependencias de datos.
- Arquitectura de experimentación: cluster de multicores.
- Se implementó inicialmente el pipeline con dos modelos de comunicación diferentes: pasaje de mensajes (*PM*) y memoria compartida (*MC1* y *MC2*).

Conclusiones

- Pensando en los diferentes niveles de memoria y en la red de interconexión de la arquitectura, se implementaron distintas alternativas (*H1*, *H2*, *H3* y *H4*) usando como modelo de comunicación un híbrido que combina pasaje de mensajes con memoria compartida.
  - *H1* y *H2* emplean un esquema de pipeline de pipeline.
  - *H3* y *H4* emplean un esquema de pipeline formado por procesos e hilos.

Conclusiones

- Se analizó el comportamiento de estos algoritmos híbridos respecto a *PM* utilizando cuatro y ocho hojas completas de la arquitectura.
- Como resultado:
  - *PM* logra una mayor eficiencia que los híbridos.
  - Diferencia amplia respecto a *H1* y *H2*.
  - Diferencia mínima respecto a *H3* y *H4*.

Agenda

- Motivación
  - Arquitecturas paralelas
  - Evaluación de sistemas paralelos
  - Principios para el diseño de algoritmos paralelos
  - Bioinformática
- Trabajo experimental / Resultados
  - Soluciones
  - Experimentos realizados y sus resultados
- Conclusiones
- Líneas de trabajo futuras

Líneas de trabajo futuras

- Analizar la escalabilidad del problema estudiado asegurando un determinado nivel de eficiencia.
- Estudiar el comportamiento de la aplicación al utilizar ambientes heterogéneos, donde se combine el cluster de multicores empleado en este trabajo con clusters heterogéneos tradicionales.
- Analizar y optimizar soluciones híbridas para determinadas clases de problemas, especialmente para los que admiten una solución paralela compuesta (combinando más de un paradigma de interacción).

¿Preguntas?

¡Gracias!